CipherCode: A Visual Tagging SDK with Encryption and Parameterisation

Ramone Karodia, Steven Lee, Ashish Mehta, Audrey Mbogho
Department of Computer Science
University of Cape Town
South Africa
{rkarodia, slee, amehta, ambogho}@cs.uct.ac.za

Abstract

Cell phones now pack as much computing power as the high-end desktop of a decade ago. This makes non-trivial mobile applications possible. Interacting with them is, however, a significant challenge due to the small size and limitations of the keypad. Because many new phone models are equipped with an integrated camera, there is growing interest in the use of visual tags to more easily provide inputs to mobile applications. A number of tag designs have been proposed, but more needs to be done in this yet nascent technology. We present CipherCode, a visual tagging system designed for speed, accuracy and compactness. CipherCode includes encryption and parameterisation, features not found in most other visual tagging schemes. We have made CipherCode freely available on the Web as an open-source SDK in order to ignite broader interest and participation in creating highly usable visual tags.

1. Introduction

Mobile phones have truly revolutionized the way we communicate and have proved to be an invaluable tool in today's hi-tech world. The mobile phone possesses qualities which make it a popular tool in many facets of life including business, fashion and entertainment. Processor speeds higher than 200 MHz are now common in high-end cell phones. This makes non-trivial mobile applications possible. However, due to the small and compact design of these devices, the degree of usability remains low and input via small keypads remains a major problem. Innovations such as predictive text and voice commands have improved this but the situation is still far from ideal. Specifically, a user risks discomfort and even injury. The end result is that much mobile potential lies dormant as applications go uninstalled or unused.

This has sparked interest in visual tags as a means of alleviating some of these usability issues affecting mobile phones.

Visual tags are small 2D labels placed on objects and captured by a camera that is attached to a computer (or phone). A tag can be a physical paper label or it can be electronically generated and displayed on an object. The labels encode information in a manner similar to barcodes, but at greater capacity. Indeed, sometimes visual tags are referred to as barcodes but it seems this is a misnomer, since squares, dots and other shapes are typically used rather than bars. Therefore any use of the term barcode in this paper is reserved for visual tags that use bars (elongated rectangles) to encode information. Otherwise, the focus is 2D visual tags.

Once an image is captured the computer performs image analysis in order to locate and decode the tag. The decoded information can be a URL [10]; credentials for connections to P2P networks [12]; a summary of the contents of a box for purposes of inventory. In other cases the decoded information can be used to manipulate a user interface [7].

A number of factors taken together create an environment that is highly conducive to the use of visual tags:

- There is increased availability of camera phones at reasonable cost. For many users, the cameras integrated into cell-phones may be just a toy accessory that is seldom used, as they will usually already own a separate "real" camera.
- Reasonably fast processors in small devices allow for resource hungry image processing tasks.
- The difficulty of providing keyed-in input to small devices has motivated exploration of alternative modalities.

- Wireless access protocol (WAP) enables information download from the Internet via small devices.

For interactive, real-time applications, there are three main requirements:

- Visual tags must seek to minimise space and at the same time maximise the amount of information they can carry. Multiple tags degrade performance which adversely affects usability in [12].
- Visual tags must maximise the proportion of time that they are read correctly.
- Visual tags must minimise the time between image capture and completion of tag decoding.

Both our tag design and decoding algorithm choices were made with the above goals in mind.

In the remainder of this paper, we first describe related work. The next section gives a summary of the key contributions of our work. We then describe the implementation details and experimental results. We also touch on our investigation of the role of image enhancement. Finally we conclude with a few remarks, including future work.

2. Related Work

The idea of using markers to aid in vision tasks is not new. Barcodes on commercial products are nothing new. Neither are fiducials (markers), which are an established part of virtual and augmented reality research for some time. What is new is the resurgence of interest (for example, [8, 10, 11, 13,]) in these markers, or visual tags, or visual codes, owing to the emergence of camera phones that are also computers. These small computers are quite powerful for their size and can manage demanding vision tasks involving small images. This capability can be exploited to address the difficulty of providing input to small devices. Instead of awkward and sometimes injurious keying in of information on the keypad, the small device rather "sees" its input. A number of interesting tags and applications have been reported. A few of these are reviewed next.

Semacode [10] is perhaps the most publicised application of visual tags. It uses an the existing standard symbology known as Data Matrix, which is a matrix made up of small black and white squares, or modules. A black module represents a binary one and a white module represents a binary zero. Creation of a Semacode tag involves embedding a URL into the tag. Decoding results in recovery of the URL so that the web page it represents can be loaded.

CyberCode [6] and the Rhos tag [8] are not based on any standard, but are similar to Semacode in that they are made up of a rectangular array of black and white modules.

The TRIP [2], SpotCode [9] and ShotCode[11] tags are circular tags made up of black and white segments. TRIP tags were used with PCs and webcams in sentient computing applications. SpotCodes were used in Bluetooth device discovery.

A coloured tag with triangular modules was proposed by Dell'Acqua et al. [3] and used in virtual reality.

While each tag is usually associated with a different set of applications, there is nothing to prevent its use in the applications reported for other tags. The underlying scheme is the same for all tags, which is the graphic representation of the digital encoding of information. Therefore, the important thing is not so much how a tag is used, but rather how it affects an application's performance.

3. Key Contributions

Our work adds to the research in the following ways:

1. We present a new compact tag design that allows for fast and accurate decoding.
2. We describe a new combination of image processing procedures for tag isolation and decoding. A variety of such procedures have been proposed. Each combination carries a different set of implications for the speed and accuracy of the overall system. It is worthwhile to explore new combinations and evaluate them against each other in order to find one that optimises performance. We take a first step in this direction.
3. We present the tagging system in the form of a freely available open-source SDK in order to encourage others to take part in this research [1].
4. Our SDK includes encryption and parameterisation in order to provide flexibility in the type of information that a tag can store.
5. Preliminary experimental results show that CipherCode outperforms Semacode [10] in speed, accuracy and compactness.

4. Implementation

In order to make our SDK as flexible as possible the system was broken down into a number of modules, each of which can be customized or replaced. The core modules are:

- Tag generator module: This essentially creates tags which encodes information which a user provides.
- Image enhancement module: This enhances images of tags that are taken with a camera in poor illumination, so as to improve the performance of the tag decoder in terms of accuracy without negatively affecting speed.
- Tag decoder module: This module decodes the information stored within a tag.

Together these modules provide the main functionality needed to successfully create and decode tags. Additional optional modules are also included which enhance the capabilities of the system. These include the Encryption/Decryption Module, the Error Correction/Detection Module, the Camera Interface Module and the ASCII/Binary Converter.

4.1. Tag Design

The **Tag Creator module** is responsible for creating the CipherCode tag. The only input to this module is an array of bits that need to be embedded into the tag. Each bit is represented by a square; black squares representing the value '1' and white squares representing the value '0'.

Figure 1. CipherCode Tag Design

Our tag (which is inspired by the Rhos tag [8], but differs considerably from it) consists of two guide bars and three cornerstones. The top guide bar is always four blocks shorter than the length of the entire tag. If the tag dimensions were 10 x 10, the top guide bar would be made up of 6 blocks. The bottom guide bar is always half the length of the top guide bar. It is oriented in such way that the centre of the bottom guide bar is inline with the centre of the top guide bar. The three cornerstones are situated in the top left, top right, and bottom left corners of the tag. The black square at the bottom right corner of the tag in Figure 1 signifies the data bits are encrypted. If there is no black square in that corner then this signifies the data bits do not require decryption. The geometry of the tag requires the tag decoder to find two parallel bars, one bar half the length of the other, with cornerstones situated collinearly with respect to the guide bars at specific relative positions. For decoding simplicity, the guide bars and cornerstones are separated by white space with respect to each other and the data area of the tag. The area in-between the row of white space adjacent to the top and bottom guide bars and cornerstones is reserved for data. For a tag of dimensions m x n (row by column), the data area of the tag will always be (m-4) x n. The '4' in the formula represents the four rows reserved for the guide bars, cornerstones, and white space.

In order to make the tag more versatile and robust the following optional modules are used by the Tag Generator:

The **Character Encoding module** is responsible for converting characters into ASCII binary bits and vice-versa. Each character within the input data is represented as a 7-bit ASCII character. In the interests of compactness, we chose not to use Unicode.

The Error **Detection and Correction module** is responsible for implementing the Hamming Code algorithm. This algorithm provides 1-bit error correction and 2-bit error detection within each block of data. Typically a block of data will be seven or eight bits.

The **Encryption module** is responsible for encrypting data before it is embedded within a tag, and decrypting data read from a CipherCode tag. This module implements the Advanced Encryption Standard [14] (AES) algorithm for encrypting and decrypting data. A 128-bit secret key is generated from a user-defined passphrase every time data is encrypted and decrypted.

The **Parameterisation module** can be used when multiple data items are required to be embedded within a visual tag. An example of this would be in creating an ID card. The tag could encode the owner's name, home address, and telephone number. Each data item would be separated by a splitter and would be decoded separately by the tag decoder. It is up to the tag developer to decide what splitter to use.

4.2. Tag Decoding

Background research revealed that many tag decoders have common designs and share numerous components. Based on these commonalities, 7 key stages were identified and integrated into the decoder design.

- Grayscale Conversion: The system begins the decoding process by first converting the colour image (obtained from the Camera Interface Module) to a grayscale image. This conversion uses the ITU standard formula: $G = (222 * Red + 707 * Green + 71 * Blue)/1000$
- Image Enhancement : The decoder then optionally uses the external Image Enhancement (discussed in a later section) Module to improve the quality, contrast and clarity of the grayscale image as well as to enhance the edges of the tag in dim light.
- Binarization: The next step is binarization which thresholds the grayscale pixel values to either 0 or 1 for black and white respectively. Two thresholding algorithms, namely global thresholding and quick adaptive thresholding (presented in [16]), were implemented and tested. Experimentation revealed that quick adaptive thresholding yielded superior results for the majority of test cases. This algorithm calculates a moving average and sets a pixel to black only if it is significantly darker than this average. Otherwise the pixel is set to white.
- Region Detection: Binarization is followed by a two pass region detection algorithm which identifies

large regions of connected black pixels. The first pass labels all black pixels according to the labels of its neighbours:

- If all the neighbours have 0 labels (i.e. are all white) then the pixel is labelled with a new unique non-zero label.
- If there is exactly one neighbouring pixel with a non-zero label then the pixel is assigned the same non-zero label
- If there is more than one neighbouring pixel with a non-zero label then the pixel is assigned the smallest label and the conflict is recorded in a special equivalence data structure.

Label conflicts are resolved during the second pass which re-labels pixels according to the equivalence data structure. This data structure is a table which stores pairs of adjacent (or conflicting) labels

- Guide Bar Identification: For each region identified in the previous step, the second-order moments [8, 15] are calculated. From these moments the eccentricity (measure of how long a region is) and orientation are calculated. Pairs of parallel and elongated regions (eccentricity greater than 6) where one bar is twice the length of the other are identified as candidate guide bar pairs
- Cornerstone Detection: The orientation and size of these guide bars pairs are then used to estimate the position of the three cornerstones. Since second-order moments provide the major and minor axis, the lengths of the bars as well as the lengths of a single cell are known. These lengths are then used to estimate the position of the cornerstones relative to the centres of the two guide bars.
- Projective Matrix Transformation: The positions of the second shorter guide bar together with the positions of the 3 cornerstones are then used in a texture mapping technique described in [4] to calculate the transformation matrix. Once this matrix is known, tag coordinates can be converted into image coordinates. The image coordinates estimates the centres of the corresponding block.
- Decoding: The final step in the decoding process is simply checking whether or not the pixel values at the calculated positions are black or white. An array is then built up and passed onto other modules for further processing.
- Other Modules: The decoder uses additional modules to convert the binary information into human readable text.

    - Error Detection/Correction: used to compensate for any decoding errors.
    - Encryption/Decryption module: If the encryption check cornerstone is filled then this module is used to acquire a key from the user and decrypt the text.
    - ASCII/Binary Converter: converts the decoded and possibly decrypted bits into ASCII characters

4.3. Experiments

Rotation Tests: these were conducted using a sample set of specially chosen images. The sample set consisted of 8 instances of the same image, each one differing by a rotation of 45°.

Tilting Tests: Tilting tests were conducted in order to determine the tilting angles at which decoding would fail. During implementation, testing revealed that perspective distortion prevented accurate calculation of tag dimensions when tilting angles were large. However, if the tag dimensions are fixed the decoder works at larger tilt angles. The decoder was, therefore, modified to be able to decode both fixed and dynamic tag sizes.

Below are shown the experimental results for decoding efficiency (Table 1) and robustness to tilting (Table 2). Compactness results for CipherCode vs. Semacode are also given in Table 3.

Type of Image	CipherCode	Semacode
Close-up of tag	4.5 seconds	6.5 seconds
No tag with few regions	5.1 seconds	10.3 seconds
No tag with many regions (complex scene)	14.3 seconds	22.1 seconds

Table 1. Decoding Speed

Tag Dimensions	Maximum Tilting Angle
Fixed	45°
Dynamic	20°

Table 2. Tilting

Chars	Semacode	CipherCode
23	Tag size: 20 by 20 Data area: 324 sq	Tag size: 20 by 20 Data area: 320 sq
28	Tag size: 22 by 22 Data area: 400 sq	Tag size: 21 by 21 Data area: 357 sq
37	Tag size: 24 by 24 Data area: 484 sq	Tag size: 24 by 24 Data area: 480 sq
40	Tag size: 26 by 26 Data area: 576 sq	Tag size: 26 by 26 Data area: 572 sq
77	Tag size: 36 by 36 Data area: 1024 sq	Tag size: 32 by 32 Data area: 896 sq

Table 3. Compactness

The results in Table 3 show that the dimensions of both tags are almost identical until a certain number of data bits are encountered. The test results for encoding strings of length 23, 37 and 40 are almost identical. The only difference is in the data area; the CipherCode tag has a slightly more compact data area. The test results for 28 characters show one of the strengths of the CipherCode tag. Semacode tags are required to have an even number of rows and columns. There is no such requirement for CipherCode tags. This enabled the CipherCode tag to embed the same website URL in a smaller odd-numbered dimension. The test results for the last case, 77 characters, are the most interesting (Figure 2). It was discovered that Semacode tags are broken up into quadrants once the data bits cannot fit into a tag of dimensions 26 x 26. This impacts the compactness of the Semacode tag. The CipherCode tag was found to be 11.11% smaller with respect to the dimensions of the tags. This equates to a reduction in the number of rows and columns by four. The data area of the CipherCode tag is also significantly smaller than the Semacode tag, sitting at 12.5%.

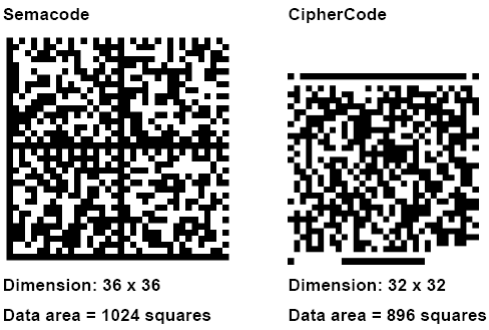

Figure 2: Encoding 77 characters , namely, "http://www.google.co.za/search?hl=en&q=visual+tag&btnG=Google+Search&meta=”

4.4. Image Enhancement

In order to increase usability in adverse conditions, we evaluated three image enhancement techniques for images captured in poor illumination. These techniques were HE (histogram equalisation), AHE (adaptive histogram equalisation), and CLAHE (contrast limited adaptive histogram equalisation). Ten pictures were taken at 4 different levels of indoor lighting, giving a sample size of 40. The results given below (Table 4) are for the results considered most significant, which were those for mid-level lighting (10 pictures). They clearly show that CLAHE is superior. We also found, however, that on cell phones, with their limited processing power, enhancing tags before decoding them slowed the application down considerably. Thus we only recommend this intermediate step for PCs and not for cell phones or other small hand-held devices until the technology improves. We also note that enhancement only makes a difference in mid-level illumination. In very bad lighting, tag decoding fails whether or not enhancement is used. Similarly, in good to excellent lighting, the decoding is highly successful without enhancement, and is not improved when enhancement is used.

Enhancement Technique	Avg. processing time (secs)	Accuracy (%)	Efficiency (rank)
HE	1.44	60	2
AHE	1.44	30	3
CLAHE	1.46	70	1

Table 4. Decoding Accuracy in Poor Light

5. Conclusions and Future Work

We hypothesise that the imaging procedures that are chosen and how they are combined affects the speed and accuracy of a visual tag-based interaction. Each tag that is devised comes with its own unique decoding algorithm, although the building blocks of each such algorithm tend to take the form of well-known image processing routines. The difference lies in which building blocks are chosen, the tasks for which they are chosen and perhaps also the order in which they are linked together.

A major contribution of our work is that we have tried out a new set of image processing procedures. From the preliminary experiments, we have obtained positive results for speed and accuracy, as well as compactness. In all three aspects, our SDK outperforms Semacode, perhaps the best known commercial system, and the only other one that, to our knowledge, is packaged as an SDK.

It is desirable to offer one approach that developers of tag-based applications can easily adapt. An early attempt to produce a general tag reading approach was presented by Ottaviani et al. [5]. It would be beneficial, however, to produce such an approach on the basis of an evaluation of all existing approaches, many of which have only recently been proposed. In further work, we shall continue to try out different methods with the ultimate goal of achieving high accuracy at real-time, interactive speeds, and we shall aim to be able to decode smaller tags.

6. Acknowledgements

This work is supported in part by the University Research Committee (URC), University of Cape Town.

References

[1] CipherCode: http://shenzi.cs.uct.ac.za/~honsproj/2006
[2] D. L. de Ipina, P. Mendonça, and A. Hopper. TRIP: A Low-Cost Vision-Based Location System for Ubiquitous

Computing. *Personal and Ubiquitous Computing Journal, 6*(3)*,* Springer*,* 2002, pp 206 – 219.

[3] A. Dell'Acqua, M. Ferrari, M. Marcon, A. Sarti and S. Tubaro. Colored Visual Tags: A Robust Approach for Augmented Reality. *Proc. IEEE Int. Conference on Advanced Video and Signal-Based Surveillance (AVSS2005)*, Como, Italy, September 2005, pp: 423 – 427.

[4] P. S. Heckbert. *Fundamentals of Texture Mapping and Image Warping*. Master's Thesis, Dept. of Electrical Engineering and Computer Science, University of California, Berkeley, 1989.

[5] E. Ottaviani, A. Pavan, M. Bottazzi, E. Brunnelli, F. Caselli and M. Guerrero. A Common Image Processing Framework for 2D Barcode Reading. *Proc. IEE International Conference Image Processing and its Applications*, Manchester, 1999, Publication Number 465, pp 652 – 655.

[6] J. Rekimoto and Y. Ayatsuka, CyberCode: Designing Augmented Reality Environments with Visual Tags. *Proc. ACM Conf. on Designing Augmented Reality Environments*, Elsinore, Denmark, 2000, pp 1 – 10.

[7] M. Rohs. Visual Code Widgets for Marker Based Interaction. *Proc. 25th IEEE international Conference On Distributed Computer Systems Workshops (ICDCSW'05),* 2005

[8] M. Rohs. Real World Interaction with Camera Phones. *Proc. 2nd International Symposium on Ubiquitous Computing Systems (UCS 2004)*, Tokyo, Japan, November 2004, pp 39 – 48.

[9] D. Scott, R. Sharp, A. Madhavapeddy and E. Upton. Using Visual Tags to Bypass Bluetooth Device Discovery. *Mobile Computing and Communications Review, 9*(1), ACM Press, 2005, pp 41 – 53.

[10] Semacode: http://semacode.org/

[11] Shotcode: http://www.shotcode.com

[12] F. Siegemund, M. Haroon and G. Brasche. Towards Pervasive Connectivity in Mobile Computing Settings. *Proc. of MPAC '06*, ACM Press.

[13] S. Siltanen and J. Hyvaka. Implementing a Natural User Interface for Camera Phones Using Visual Tags. *Proc. 7th Australian User Interface Conference (AUIC2006)*.

[14] W. Stallings. *Network Security Essentials – Applications and Standards*, 2nd Edition, Prentice Hall, 2003.

[15] R. C. Veltkamp and M. Hagedoorn. State of the Art in Shape Matching. M. Lew (Ed.), *Principles of Visual Information Retrieval,* pp. 87-119, Springer.

[16] P. Wellner. Interacting with Paper on the DigitalDesk. *Communications of the ACM, 36*(7), 1993, pp 87 – 96.